INFO-F-404: Real-Time Operating Systems

2017 – 2018 Project 1: Audsley

1 Main goal

Study Audley's priorities assignment algorithm for asynchronous constrained deadlines tasks sets. We will consider systems of n periodic, asynchronous and independent tasks $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ with constrained deadlines on a uniprocessor platform.

2 Project details

Audsley's algorithm is based on the theorem stating that, if a task τ_i from the tasks set τ is lowest-priority viable, there exists a feasible FTP-assignment for τ if and only if there exists a feasible FTP-priority assignment for $\tau \setminus \{\tau_i\}$.

For this project, tasks sets will be described in text files. Each line of a given file describes one task and contains: `Offset Period Deadline WCET`.

Here is an example:

```
0 50 50 10
0 80 80 20
0 100 100 10
0 200 200 50
```

For this project, we ask you to realise the following parts:

Feasibility interval
For the considered systems, $[0, S_n+P_n)$ is a feasibility interval. However, to use this feasibility interval, one shall already have assigned priorities to tasks. $[O_{\max}, O_{\max} + 2 \cdot P]$ is a feasibility interval that does not need preexistent priorities assignment. You are asked to implement a script that, for a system given as a parameter in a file, prints the feasibility interval $[O_{\max}, O_{\max} + 2 \cdot P]$. For example, with the system seen in the exercices session about Audsley on the file `audsley.txt`, the following call:

```
python project.py interval audsley.txt
```

shall print: `100,400`

Simulator
You are asked to create a script that implements a single processor FTP simulator that simulates the system for a given period. The start and stop points of the simulation are given as command line parameters, as well as the file containing the task set. The tasks are ordered by decreasing

priorities in the file.

You should be able to execute your program using the following command line:

```
python project.py sim <start> <stop> <tasksFile>
```

for example, with the file described above, the following call:

```
python project.py sim 0 200 tasks.txt
```

will produce the following output:

```
Schedule from: 0 to: 200 ; 4 tasks
0: Arrival of job T1J1
0: Arrival of job T2J1
0: Arrival of job T3J1
0: Arrival of job T4J1
0-10: T1J1
10-30: T2J1
30-40: T3J1
40-50: T4J1
50: Arrival of job T1J2
50: Deadline of job T1J1
50-60: T1J2
60-80: T4J1
80: Arrival of job T2J2
80: Deadline of job T2J1
80-100: T2J2
100: Arrival of job T1J3
100: Arrival of job T3J2
100: Deadline of job T1J2
100: Deadline of job T3J1
100-110: T1J3
110-120: T3J2
120-140: T4J1
150: Arrival of job T1J4
150: Deadline of job T1J3
150-160: T1J4
160: Arrival of job T2J3
160: Deadline of job T2J2
160-180: T2J3
200: Deadline of job T1J4
200: Deadline of job T3J2
200: Deadline of job T4J1
```

This schedule does not show a deadline miss. In case of a deadline miss, the output is as

follows:

```
timeInstant: Job TxJy misses a deadline
```

Lowest-priority viable
You are asked to implement a function that, for a given system, interval and a specified task number, returns `True` if the task is lowest-priority viable, `False` otherwise.

Audsley's algorithm
You are asked to implement a script that, for a system and an interval given as parameters, depicts the research made by the Audsley's algorithm to find a schedulable priorities assignment. Normally, Audsley's algorithm will stop the research when such an assignment is found. However, here, you are asked to continue the research to present all possible assignments that may be found using the technique. The level of recursion will be showed on screen using spaces. For example, a call to the script with the system seen in the exercices session about Audsley on the file `audsley.txt`:

```
python project.py audsley 0 400 audsley.txt
```

shall produce the following output:

```
Task 1 is not lowest priority viable
Task 2 is not lowest priority viable
Task 3 is lowest priority viable
 Task 1 is not lowest priority viable
 Task 2 is lowest priority viable
  Task 1 is lowest priority viable
```

This output shows that, for this tasks set, the only feasible FTP priorities assignment is given by $\tau_1 \succ \tau_2 \succ \tau_3$.

Schedule plotter You are asked to write a program that generates a visual output of the scheduling. The choice of the graphical library is up to you but it shall be available freely. You may choose the format (pdf, png, bmp, avi, *etc.*).

Generator Implement a generator of random periodic, asynchronous systems with constrained deadlines. This generator should be able to generate a system with given parameters (utilisation factor and number of tasks), for example:

```
python project.py gen 6 70 tasks.txt
```

have to generate a file `tasks.txt` that describes a system of 6 tasks with an utilisation of 70% (utilisation of the generated system does not have to be equal to 70% but should be very close to it).

Report
Write a short report that contains at least:

- a description of your code (diagrams) and implementation choices,
- a section where you describe difficulties that you met during this project (and solutions that you found),
- a section stating the library you used for the graphical part, where it can be downloaded and how it should be installed.

The project has to be implemented using *python* programming language.

You are allowed to add more additional (optional) command line parameters to your programs.

3 Submission and planning

This project should be done in groups of 3 maximum, you may choose your partner(s). This project has to be submitted before 23:55 on the 21^{th} of December of 2017. Your project has to work properly under *Linux* in rooms NO3.007, NO4.008 and NO4.009 (ULB, Plaine) except for the graphical part that may require installation of additional software.

You'll submit a folder in a *zip* file that contains at least the following files:

- your python scripts,
- a short report (pdf format, maximum 5 pages plus an appendix if needed, figures and graphics are welcome).

The zip file with your project has to be submitted to the UV: `http://uv.ulb.ac.be`. The name of the folder and of the zip file will be as follows: if Jean Dupont made his project with Albertine Vanderbeken, they should send a file named *dupont-vanderbeken.zip* (that contains a folder of the same name, that contains all your project files).

Please observe that if your project crashes during execution or the report is missing or it is not submitted following the described guidelines then your project will not be graded.

For questions on this project, please refer to **cedric.ternon@ulb.ac.be** with subject "[INFO-F-404] project1".

Good luck!